## REMARKS

Applicant has amended claims 1, 8, and 14 to address the shortcomings of the prior art. More specifically, the amendments presented herein represent the novel aspect of determining compatibility of a software version on a node joining a cluster when target data stored in a shared storage resource in the cluster is formatted by the same or an alternate software version. Applicant's invention is directed to determining compatibility of the software version on the node joining the cluster with the data structure in shared cluster storage to be accessed by that software version. Applicant's invention is not directed at determining compatibility or operation of a software version with any other software versions present in the cluster. The nodes of the cluster of Applicant may operate under different software versions, wherein each node may continue to communicate with shared data storage using the version control record to insure that the shared data is compatible with the software version accessing that data from the member node. Accordingly, Applicant's invention pertains to controlling a joining node's entry to the cluster based upon the compatibility of the software version present on the joining node with the target data present on the shared storage.

*Short et al.* is limited to software compatibility to various dynamic-link libraries, or DLLs, utilized by a software program present on the various nodes of the cluster. See Exhibit A. *Short et al.* provides a method of controlling use of a software version of the software program on a node of a cluster through determining compatibility of that software version with the DLLs stored on the cluster and required for operation of that software, irrespective of version. Thereby, the method of *Short et al.* insures that a software version attempting to utilize a required DLL is compatible with the DLL prior to allowing the software version to utilize the DLL. In this manner, *Short et al.* provides a method of preventing operation of a software version on a node in a cluster when the DLLs for the software program stored on the cluster are not compatible with the specific software version attempting to execute. Accordingly, *Short et al.* provides a method of preventing execution of a software program when the required program components, in the form of the DLLs, are not present or accessible.

The prior art reference of *Short et al.* does not address accessibility to target data. *Short et al.* is limited to the cluster level comparison of software versions to determine a software version's ability to execute utilizing various DLLs stored on the cluster and required to allow operation of the software application, *i.e.* a required component of the software application as in its DLLs. *Short et al.* does not teach or suggest determining compatibility of a software version present on a joining node with that software application's target data stored in the cluster. In other words the compatibility determination of *Short et al.*, which is directed to the ability of the software version to execute on the cluster, is not equivalent to the compatibility determination as claimed by Applicant, which is directed to the ability of a software version to access the target data created by a software application, albeit a different version of the software application. No new matter has been added to the application with the amendments to the claims submitted herein. Support for the amendments is found in paragraphs 0005-0006 and 0022 - 0025.

In view of the foregoing amendment to the claims, it is submitted that all of the claims remaining in the application are now in condition for allowance, and such action is respectfully requested. Applicant is not conceding in this application that those claims in their prior forms are not patentable over the art cited by the Examiner, as the present claim amendments are only for facilitating expeditious prosecution of the application. Applicant respectfully reserves the right to pursue these and other claims in one or more continuation and/or divisional patent applications. Should any questions arise in connection with this application or should the Examiner believe that a telephone conference with the undersigned would be helpful in resolving any remaining issues pertaining to this application; the undersigned respectfully requests that she be contacted at the number indicated below.

For the reasons outlined above, withdrawal of the rejection of record and an allowance of this application are respectfully requested.

Respectfully submitted,

By: /Rochelle Lieberman/
Rochelle Lieberman
Registration No. 39,276
Attorney for Applicant

Lieberman & Brandsdorfer, LLC
802 Still Creek Lane
Gaithersburg, MD 20878
Phone: 301-948-7775
Fax: 301-948-7774
Email: rocky@legalplanner.com

Date: October 22, 2008

EXHIBIT A





# Dynamically Linked Library



## 1 dictionary results for: *Dynamically Linked Library*

*Free On-line Dictionary of Computing* - *Cite This Source* - *Share This*

**Dynamically Linked Library** *library*

(DLL) A library which is linked to application programs when they are loaded or run rather than as the final phase of compilation. This means that the same block of library code can be shared between several tasks rather than each task containing copies of the routines it uses. The executable is compiled with a library of "stubs" which allow link errors to be detected at compile-time. Then, at run time, either the system loader or the task's entry code must arrange for library calls to be patched with the addresses of the real shared library routines, possibly via a jump table.

The alternative is to make library calls part of the operating system kernel and enter them via some kind of trap instruction. This is generally less efficient than an ordinary subroutine call.

It is important to ensure that the version of a dynamically linked library is compatible with what the executable expects.

Examples of operating systems using dynamic linking are SunOS (.so - shared object files), Microsoft Windows (.dll) and RISC OS on the Acorn Archimedes (relocatable modules).

(1995-12-12)

EXHIBIT A



# Dynamically Linked Library